

**FILE COPY**

ESD-TR-76-357

~~ESD ACCESSION LIST~~

DRI Call No. 88516

Copy No. 1 of 1 cys.

PROTOTYPE SECURE MULTICS SPECIFICATION

Honeywell Information Systems, Incorporated  
Federal Systems Operations  
7900 Westpark Drive  
McLean, VA 22101

January 1976

Approved for Public Release;  
Distribution Unlimited.

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS  
ELECTRONIC SYSTEMS DIVISION  
HANSCOM AIR FORCE BASE, MA 01731



ADA055164

### LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

### OTHER NOTICES

Do not return this copy. Retain or destroy.

This technical report has been reviewed and is approved for publication.

William R Price  
WILLIAM R. PRICE, Captain, USAF  
Techniques Engineering Division

Donald P. Erikson  
DONALD P. ERIKSEN  
Techniques Engineering Division

FOR THE COMMANDER

Stanley P. Dereska  
STANLEY P. DERESKA, Colonel, USAF  
Deputy Director, Computer  
Systems Engineering  
Deputy for Command & Management Systems

## PREFACE

Because of funding limitations, the Air Force terminated the effort which this document describes before the effort reached its logical conclusion. This specification has not been formally approved but was published in the interest of capturing and disseminating the computer security technology that was available when the effort was terminated.

This specification was to describe the characteristics of a secure general purpose computer system (namely Multics) based upon a security kernel. A previous Air Force sponsored project to develop security controls for an operational Air Force computer system served as a background and starting point for the effort that was to be described here. However, the Air Force terminated the effort to be reported before the results could be documented and, consequently, much of the information found in this report is drawn from the documented results of the previous project.\*

\*Whitmore, J. et al, "Design for Multics Security Enhancements" Honeywell Information Systems, Inc., ESD-TR-74-176, December 1973.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER <b>ESD-TR-76-357</b>	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) <b>PROTOTYPE SECURE MULTICS SPECIFICATION</b>		5. TYPE OF REPORT & PERIOD COVERED
7. AUTHOR(s)		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS <b>Honeywell Information Systems, Incorporated Federal Systems Operations 7900 Westpark Drive, McLean, VA 22101</b>		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS <b>Deputy for Command and Management Systems Electronic Systems Division Hanscom Air Force Base, MA 01731</b>		12. REPORT DATE <b>January 1976</b>
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		13. NUMBER OF PAGES <b>42</b>
16. DISTRIBUTION STATEMENT (of this Report)  <b>Approved for Public Release; Distribution Unlimited.</b>		15. SECURITY CLASS. (of this report) <b>UNCLASSIFIED</b>
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE <b>N/A</b>
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)  <b>Security, Security Kernel, Multics, Multilevel Security</b>		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)  <p>The goal of Project Guardian is to design, develop and certify a secure Multics to provide a certified secure multilevel computer utility. This report covers preliminary work in development of a specification describing the characteristics of the secure system.</p>		

**SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)**

PROJECT GUARDIAN

PROTOTYPE SECURE MULTICS  
SPECIFICATION

Preliminary Draft

January 31, 1976

prepared for

Department of the Air Force  
Electronic Systems Division  
Hanscom Air Force Base  
Bedford, Massachusetts 01731

Contract No. F19628-74-C-0193

CORL Sequence No. A013

Honeywell Information Systems, Inc.  
Federal Systems Operations  
7900 Westpark Drive  
McLean, Virginia 22101

## Contents

### Introduction

#### 1.0 Scope

##### 1.1 Identification and Authority

##### 1.2 Background

##### 1.3 Purpose

#### 2.0 Applicable Documents

#### 3.0 Functional Requirements for a Secure Multics

##### 3.1 Definition of Concepts and Terminology

##### 3.2 Overview of System Architecture

###### 3.2.1 Hardware Configuration

###### 3.2.2 Internal/External Environment

###### 3.2.3 The Security Kernel

###### 3.2.4 The Secure Front-End Processor

###### 3.2.5 The System Security Perimeter

##### 3.3 Process Creation

###### 3.3.1 Logging Into the System

###### 3.3.2 User Authentication

###### 3.3.3 Process Clearance Assignment

##### 3.4 The Multics Storage System

###### 3.4.1 Access to Segments

###### 3.4.2 Access to Directories

## Contents (continued)

- 3.5 Use of I/O Devices
  - 3.5.1 Internal I/O
  - 3.5.2 External I/O
  - 3.5.3 Bulk I/O services
- 3.6 Interprocess Communication
  - 3.6.1 Block/Wakeup Mechanism
  - 3.6.2 Message Segments
- 3.7 Trusted Processes and System Functions
  - 3.7.1 System Control Process
  - 3.7.2 Backup
  - 3.7.3 Retrieval
  - 3.7.4 System Security Administrator
  - 3.7.5 System Administrator
  - 3.7.6 Maintenance and Repair Processes
  - 3.7.7 The I/O Coordinator Process
- 3.8 System Audit
- 4.0 Quality Assurance
- 5.0 Preparation for Delivery
- 6.0 Notes

## Introduction

Several years ago, the Air Force identified a need for a general purpose computer system which could securely protect all information it contained. The two general requirements were: access controls which would support the military security system; and certification that the access controls could not be defeated by a malicious user or by accidental failure. Honeywell has been working with the Air Force to develop this secure system based on the architecture of Honeywell's Multics system.

The first task was to design the access controls to support the military security system. This task has been completed and resulted in the inclusion of the Access Isolation Mechanism in the Multics standard product.

The next task, certifying the correctness of the access controls, is a much larger effort. This report attempts to provide an initial description of a prototype secure Multics system capable of meeting some certification criteria. The description is preliminary and is expected to change as future design efforts provide further insight into the problems of certification.

The modifications described in this report for the development of a prototype secure Multics system are subject to formal approval by both Honeywell and the Air Force before they become binding on further development efforts.

## 1. SCOPE

### 1.1 Identification and Authority

The authority for this specification is contained in contract number F19628-74-C-0193. The specification corresponds to CDRL sequence number A013.

### 1.2 Background

The problem of security in computer systems has been under study for several years. The Air Force has sponsored several studies and development projects aimed at improving understanding of security in computer systems, developing a sound theoretical basis for further work, and demonstrating accomplishments in the field. Many of these projects have been associated with the Multics system.

The overall goal of these efforts has been to develop a certifiably secure computer system for general use by the military to meet their operational requirements. This report is part of an effort to take the Multics system from its present form to a prototype Multics system which can be used to demonstrate the feasibility of software certification.

The military faces an increasing need for operational computer systems capable of processing several levels of classified information at the same time. Present systems are unable to support secure multilevel processing due to fundamental weaknesses in their basic design, since security was not a concern when they were developed. The weakness is that current hardware/software systems are unable to adequately protect the information that they process.

Currently, the military meets the need for processing several levels of information by one of two methods. Either all security levels are processed together at the level of the highest classification present, or each level is processed by itself. Both methods have been less than satisfactory. The problem with processing all levels

together is that all users and all equipment, including terminals and communications facilities, must be cleared to the highest classification that the system can ever process. The problem with separate processing is that a separate computer system or a separate period of time is required for each level handled. Either method is costly and inefficient. Neither method allows simultaneous handling of information at several levels for users of several levels of clearance.

Multics is the most advanced general utility system as far as security is concerned. Security was one of the initial design goals of the Multics system designers and has been a major concern of the designers and developers throughout the history of the system. Even with this concern for security, the present Multics system cannot be certified secure. Multics, however, does present the best available base upon which to build a certifiably secure multilevel computer utility.

Secure communications has also presented operational problems to the military. A secure on-line system requires a secure communications network. While the techniques of securing communications lines and terminals have been well developed, a certifiably secure communications processor is still undeveloped. A secure Multilevel system must have a compatible and Secure Front-End Communications Processor to be able to properly handle multiple levels of classified information.

Both economic and operational considerations make development of a certifiably secure multilevel system desirable. Recent advances in computer technology indicate that it should be possible to produce a system that can process an arbitrary mix of classified and unclassified information simultaneously on a single computer system. The system should serve both cleared and uncleared users and should rely on the computer system's internal hardware/software controls to enforce security and need-to-know requirements. Of primary importance is that the system be certifiably secure. That is, it must be possible to prove that the system is complete and without flaw in any of its security-related aspects.

The Air Force has been working on the problem of providing a certifiably secure multilevel system for several years. In 1970, the Air Force Data Services Center (AFDSC) requested the Electronic Systems Division (ESD) to support development of an open multilevel system for the AFDSC Honeywell 635 systems. The resulting studies pointed out the severity of the problem and led to the formation of a computer security

technology planning study panel. The panel's report (2.19) described the fundamental problems and delineated a program to develop the desired system. The panel recommended that the technical approach to the problem be "to start with a statement of an ideal system, a model, and to refine and move the statement through various levels of design into the mechanism that implement the model system".

The basic component of the ideal system was also identified by this panel. This component is known as the Reference Monitor, an abstract mechanism that controls access of subjects (active system elements) to objects (units of information) within the computer system and enforces the rules of the military security system on such access. Three requirements were recognized for a Reference Monitor:

- a. Complete Mediation - the mechanism must mediate every access of a subject to an object.
- b. Isolation - the mechanism and its data bases must be protected from unauthorized alteration.
- c. Verifiability - the mechanism must be small, simple, and understandable so that it can be completely tested and verified (certified) to perform its functions correctly.

The mechanism that implements the Reference Monitor in a particular computer system has been termed the security kernel. Much subsequent work has been devoted to identifying the characteristics of a security kernel and to exploring the technology involved in producing a security kernel for some computer system.

ESD initiated development of formal mathematical models of the ideal Reference Monitor in 1972. This work (2.10, 2.11, 2.13) resulted in a model of a secure computer system as a finite-state mechanism that makes explicit transitions from one secure state to another. The rules of the model formally define the conditions under which a transition from state to state can occur. The rules have been proven to allow only transitions that preserve the security of information in the system. The model specifies requirements for the operation of a security kernel. These requirements were taken directly from the Defense Department regulations on handling sensitive information (DoD Directive 5200.1-R). With the availability of the model, the problem of validation is now reduced to providing complete assurance that a particular security kernel behaves exactly as the model requires.

Work on the technology of certification progressed in parallel with the work on the model. In 1973, Price (2.21) identified a methodology for verification of a kernel. More detailed developments of this validation methodology have been reported by MITRE (2.17, 2.21). Another approach has been explored which may be more suitable to large software modules (2.23).

Other activities have been devoted to the problem of building a security kernel for a practical system (2.16, 2.24). This work has demonstrated the soundness of the basic concepts and also pointed out some of the problems that lie in the way of realizing a security kernel on a large system. This work has been the basis for development of a secure communications processor, a detailed effort which is presented later in this report.

A major project in the development process is the development of a security kernel for a large resource sharing system. The system chosen for this effort is Multics. There are two reasons that this choice was made. First, the hardware base of the Multics system, the Honeywell 68/80 computer, has been identified as best suited of all off-the-shelf large computer systems for the support of a security kernel (2.25). Second, the Multics system architecture was conceived and developed with security requirements specifically in mind.

One project, now completed, involved the design and production of a Multics system capable of supporting a two-level (Secret and Top Secret) environment for the Air Force Data Services Center (2.15, 2.26). This system implements security controls based on the military access rules, but it does not completely handle the threat of a hostile penetration. From these efforts, additional insight was gained in the problems of designing and developing a security kernel for Multics.

Design of a security kernel for Multics was started as a joint effort between personnel from ESD, the MITRE Corporation, the Massachusetts Institute of Technology, and Honeywell Information Systems. This design effort has led to a more complete understanding of the general problem. Work is progressing on formal specifications for the Multics Security Kernel (2.27) and on simplification and reorganization of the Multics operating system. Based on these efforts, the current Multics system will be refined and critical portions of the system will be reimplemented to produce a certifiable kernel which will interface with a certifiable front-end communications processor with its own security kernel. The result will be a prototype Multics system which may meet the goal of Air Force certification.

The overall project can be described in terms of three major development activities: development of a technology to support the development of a certifiable system, development of hardware and software for a prototype Secure Front-End Processor, and development of a certifiable prototype Multics system with a security kernel interfacing with the Secure Front-End Processor.

A new collection of programming methodologies and techniques is developed to support the major Multics and SFEP development activities. Included are: techniques for formally specifying software; techniques for demonstrating the correspondence among hierarchically ordered levels of specification; programming languages emphasizing the generation of certifiable code; and support tools aiding the automation of certification of specifications and code, and performance measurements.

The Secure Front-End Processor is developed using a hardware architecture designed specifically to provide a basis for certification according to the Air Force security model. The hardware provides segmented addressing and interfaces directly with the prototype Multics system. The software provides a kernel-based system architecture with a supervisor supporting communications subsystems for external I/O. The initial version of the SFEP software is integrated with the kernel-based Multics system to demonstrate functional capability. A second SFEP version is then coded in the new system programming language, incorporates performance enhancements and is then integrated with Multics.

Development of the certifiable prototype Multics system entails the restructuring of the present Multics supervisor to rely on a formally-specified security kernel leading to three demonstrable prototypes. The first demonstrable prototype employs a formally-specified kernel, coded in PL/I, interfacing with a DATANET 6600 front-end processor and establishes functional and performance measures of the design. The second demonstrable prototype incorporates the Secure Front-End Processor to demonstrate successful integration of the two hardware systems with their respective kernels and supervisors. The third demonstrable prototype incorporates performance enhancements and contains kernels coded in the new system programming languages. This last prototype establishes the feasibility of certifying code correctness aided by the support tools developed as part of the project.

### 1.3 Purpose

This document is the first step towards the specification of a certifiable secure prototype Multics system. The problems and implications of restructuring the present Multics supervisor to rely on a formally-specified security kernel are examined. The mathematical model of computer security prepared by the MITRE Corporation (2.10, 2.11) provides the theoretical criterion for evaluating design alternatives. The result is a functional description of the user interface as modified to support a kernel-based system design.

The system architecture described here will provide the basis for more detailed and formal specifications of the security kernel and revised supervisor. These will be prepared during the course of developing a prototype secure Multics system.

The form of presentation for this specification assumes a good working knowledge of the structure of the current Multics system. Readers are directed to the Honeywell documents listed in paragraph 2.7 for more information on the concepts presented.

#### 1.3.1 Specific Exclusions

Certain problems of multi-level security ADP operation and extensions of basic multi-level security controls were known at the start of the design effort and were specifically excluded. Certain system functions which do not serve to demonstrate the feasibility of the security kernel concept have been specifically excluded. These are described in the following paragraphs.

##### 1.3.1.1 The Trojan Horse Problem

A computer system which provides sharing of user written procedures is susceptible to a "Trojan Horse attack" by a malicious user (See Section 3.1.3 for definition.) A general solution to the Trojan Horse problem is excluded from the scope of the design effort. However, reducing the information paths between users of different security levels is within the scope of the design effort. The issue of sabotage from a Trojan Horse is accepted with a low expectation of occurrence in a system used for general

purpose computing. However, an act of sabotage by a Trojan Horse could have considerably more severe consequences at certain military sites such as those having a command and control environment.

#### 1.3.1.2 High-Water Mark

The design of having users start work at a low-security level with automatic or requested upgrade to a higher-security level as more sensitive data is needed was specifically excluded from the scope of this design. This is commonly described as a "high-water mark" capability.

#### 1.3.1.3 Program Trustworthiness

The ability to reduce the system recognized clearance of a user who may attempt to access sensitive material, based on the clearance level of procedures executed in a user's process, is commonly described as the "trustworthiness" capability. This is one means to reduce the potential damage by a Trojan Horse attempting to perform sabotage. This "trustworthiness" capability is specifically excluded from the scope of the design effort.

#### 1.2.2.5 Accounting, Billing and Load Control

The current accounting and billing procedures form a very large subsystem on Multics. Many interfaces to the system are involved, most of which will now contain multi-level security data. The restructure of the system will change most of these interfaces, thus requiring a large effort to make the accounting and billing procedures work on a kernel-based Multics system. No benefit in the demonstration of a prototype secure Multics system will accrue from making such modifications at this time. Therefore, accounting and billing functions are specifically excluded from this effort.

Also, the load control group mechanism is only needed for a heavily used Multics service. Only a primitive mechanism is needed to demonstrate the feasibility of a kernel-based Multics system.

## 2. APPLICABLE DOCUMENTS

### 2.1 Air Force/Honeywell contract number F19628-74-C-0193

This contract provides the authority for the design effort. The documentation requirements for the final report and the allowed deviations from the format are specified in this contract.

### 2.2 DoD 5200.1-R Information Security Program Regulation

Describes the military security system and the responsibilities of personnel who fall within its jurisdiction.

### 2.3 AFR 205-1 Information Security Program (USAF)

Implements DoD 5200.1-R

### 2.4 DoD 5200.28 Department of Defense Directive, Security Requirements for Automatic Data Processing (ADP) Systems

Defines the security requirements for processing classified data on an ADP system (See 2.5).

### 2.5 DoD 5200.28-M Manual of Techniques and Procedures for Implementing, Deactivating, Testing, and Evaluating - Secure Resource Sharing ADP Systems.

This is the manual which outlines the details of the general requirements specified in DoD 5200.28.

### 2.6 MIL-STD-483 Appendix III

Air Force suggested documentation format specification for the final report.

This standard has been followed for content and general order of presentation. Deviations from the strict format

were authorized by the contract (Paragraph 2.1). Section 3 of the standard has been expanded in this document to provide a form of presentation better suited to the material.

## 2.7 Standard Honeywell Multics documentation

The following documents are mentioned here as a source of background information concerning the Standard Multics system.

Multics Programmers' Manual  
    Introduction (AG90)  
    Reference Guide (AG91)  
    Commands and Active Functions (AG92)  
    Subroutines (AG93)  
    Subsystem Writer's Guide (AK92)  
Project Administrator's Manual (AK51)  
System Administrator's Manual (AK50)  
PL/I Language Manual (AG94)  
Multics Virtual Memory (AG95)  
The Multics System (AK27)

The order numbers given above (e.g. AG90) should be specified when ordering these documents from Honeywell.

2.8 Multics Evaluation, J.P. Anderson, ESD-TR-73-276, Electronic System Systems Division (AFSC), L. G. Hanscom Field, Bedford, MA, October 1973.

2.9 Design and Certification Approach: Secure Communications Processor, P. S. Tasker and D. E. Bell, MTR-2436, The MITRE Corporation, Bedford, MA.

2.10 Secure Computer Systems: Mathematical Foundations, D. E. Bell and L. J. LaPadula, ESD-TR-73-278, Vol I, Electronic Systems Division (AFSC), L. G. Hanscom Field, Bedford, MA, November 1973.

2.11 Secure Computer Systems: A Mathematical Model, L. J. LaPadula and D. E. Bell, ESD-TR-73-278, Vol II, Electronic System Division (AFSC), L. G. Hanscom Field, Bedford, MA, November 1973.

2.12 Computer Secure Research and Development Requirements, S. B. Lipner, MTP-142, The MITRE Corporation, Bedford, MA, February 1973.

- 2.13 Preliminary Notes on the Design of Secure Military Computer Systems, R. R. Schell, P. J. Downey, and G. J. Popek, MCI-73-1, Electronic Systems Division (AFSC), L. G. Hanscom Field, Bedford, MA, January 1973.
- 2.14 Concept of Operation for Handling I/O in a Secure Computer at the Air Force Data Services Center (AFDSC), E. L. Burke, ESD-TR-74-113, L. G. Hanscom Field, Bedford, MA, October 1973.
- 2.15 Design for Multics Security Enhancements, J. Whitmore, A. Bensoussan, P. Green, D. Hunt, A. Kobziar, J. Stern, ESD-TR-74-176, L. G. Hanscom Field, Bedford, MA 1973
- 2.16 The Design and Specification of a Security Kernel for the PDP-11/45, W. L. Schiller, ESD-TR-75-69, L. G. Hanscom Field, Bedford, MA, May 1975
- 2.17 A Software Validation Technique for Certification: The Methodology, D. E. Bell and E. L. Burke, ESD-TR-75-54, Electronic Systems Division (AFSC), L. G. Hanscom Field, Bedford, MA., April 1975
- 2.18 Primitive Models for Computer Security, K. G. Walter, W. F. Ogden, W. C. Rounds, et al Department of Computing and Information Sciences, Case Western Reserve University, Cleveland, Ohio, ESD-TR-74-117, Electronic Systems Division (AFSC), L.G. Hanscom Field, Bedford, MA., January 1974
- 2.19 Computer Security Technology Planning Study, J. P. Anderson, ESD-TR-73-51, Vol II, Electronic Systems Division (AFSC), L.G. Hanscom Field, Bedford, MA., October 1972
- 2.20 Computer Security Development Summary, ESD 2073, Electronic Systems Division, L.G. Hanscom Field, Bedford, MA., December 1973
- 2.21 Implications of a Virtual Memory Mechanism for Implementing Protection in a Family of Operating Systems, W. R. Price, PhD Thesis, Carnegie-Mellon University, June, 1973
- 2.22 Synthesis of a Software Security System, E. L. Burke, MTP-154, The MITRE Corporation, Bedford, MA.
- 2.23 On Attaining Reliable Software for a Secure Operating System, L. Robinson, P. G. Neumann, K. N. Levitt, A. Saxena, 1975 International Conference on Reliable Software, Los Angeles, Ca., April 1975
- 2.24 Design of a Security Kernel for the PDP-11/45, W. L. Schiller, The MITRE Corporation, Bedford, MA., December 1973

- 2.25 Architectures for Secure Computing Systems, L. Smith,  
ESD-TR-75-51, The MITRE Corporation, Bedford, MA., June 1974
- 2.26 Access Isolation Mechanism Pre-Release Documentation,  
Honeywell Information Systems, Inc., McLean, Va., August  
1975
- 2.27 The Top Level Specification of a Multics Security Kernel, W.  
L. Schiller, K. J. Biba, E. L. Burke, Working Paper  
WP-20377, The MITRE Corporation, Bedford, Ma., August 1975
- 2.28 Initial Structured Specifications for an Uncompromisable  
Computer Security System, K. G. Walter, W. F. Ogden, et al  
Case-Western Reserve University, Cleveland, Ohio, July 1975

### 3.0 Functional Requirements for a Secure Multics

A secure computer system is one which can successfully protect all data entrusted to it from unauthorized disclosure. This is the basic definition of system security or more specifically system software security which guides the Guardian project. Issues of physical security which can deny service to authorized users are specifically ignored here (e.g., fire, flood, etc.). The major concern is to counter all security threats which would allow someone to steal information (or data) from the computer system. The security threats of general interest fall into three logical areas: malicious persons external to the system, authorized users of the system and collusion between authorized users.

The threats from malicious persons external to the system are not particularly interesting to the system software designer. These threats include: tapping communication lines; stealing listings, tapes, terminal output or other data generated by the system; stealing passwords of authorized users; monitoring electromagnetic emanations from the hardware; or unauthorized actions by operations or administrative personnel. Each of the threats mentioned can only be countered by physical or procedural security measures external to the computer system. The only external threats of interest to the system software designer are illegal entry attempts (login) and operational errors. These are solved by the use of passwords for user authentication and by providing unambiguous instructions and/or messages to operations personnel.

The threats from collusion between authorized users is also uninteresting. The Multics Access Isolation Mechanism makes it extremely difficult for two users of different security clearances to compromise data while it remains resident in Multics, even though the current system is not certified secure. Therefore, two or more users in collusion to steal data will find it easier to pass the data external to the system; an information channel which is beyond the control of the system designer.

The remaining security threats come from persons authorized to use the system. This is the area of particular interest in this development effort. The less severe internal

threats of browsing by a curious user and accidental granting of access have been addressed by the implementation of the Access Isolation Mechanism. The insidious threats of a Trojan Horse program or system penetration remain to be solved.

Within the Multics architecture, a general solution to the threat of a Trojan Horse has not been found. However, for a Trojan Horse program to be able to compromise data, it must be able to communicate between security levels. Therefore, one requirement of this effort is to eliminate all communication paths which would allow a program to read data of one security level and write it where it could be read from a lower security level.

A user who can penetrate the operating system may be able to invalidate all the access control mechanisms. A penetration can occur from incorrect implementation of the various protection mechanisms or from a malicious programmer inserting special code sequences to provide a "trap door" into the operating system. Therefore, another requirement of this effort is to verify the correct implementation of the Multics operating system and to verify that no trap doors exist.

The Multics protection mechanisms are implemented within the most privileged protection ring, ring 0. Unfortunately, there are a large number of programs in ring 0 which are very complex. The interactions between these programs are also complex and often subtle or obscure. In addition, there are no mechanisms to protect programs and data within ring 0 from errors in other programs in this ring. Therefore, any attempt to verify the correctness of the current Multics supervisor as it exists is doomed to failure from the start.

The approach to meeting the requirements is to restructure the current Multics operating system to isolate the primitive mechanisms which implement the security access controls. This will form the reference monitor or security kernel of Multics. The mathematical model of computer security (2.10, 2.11) is the criterion used in defining the interface between the kernel and other parts of the system. Good engineering practice requires that the current operating system be molded into the new structure rather than attempting a complete top down redesign. It is expected that several iterations between top down specification for correctness proofs and bottom up design for engineering feasibility will be needed.

The following paragraphs within section 3 of this specification form the initial description of the restructured Multics system based on bottom up design.

### 3.1 Definition of Concepts and Terminology

#### 3.1.1 Hardware/Software Interface

The central processing unit used is the Honeywell series 60 level 68/80. The operating system is Multics, with such modifications and extensions as result from this design effort and the system programming task that will follow.

A full description of the Honeywell 6880 hardware and the Multics software is beyond the scope of this document. The interested reader is referred to the publications listed in Section 2.7 for such detailed descriptions.

#### 3.1.2 User Interface

The user interface is the appearance the system presents to the user. To the greatest degree possible, this appearance will remain the same as current Multics.

Functions available to the user will be identical to current Multics where feasible, and equivalent in most other cases.

#### 3.1.3 General Definitions

##### access

The ability and the means to approach, communicate with (input to or receive output from), or otherwise make use of any material or component in an ADP System.

In the military security system, a person may be granted access to an object only if his clearance level is greater than or equal to the classification level of the object; his clearance category set contains all categories in the category set of the object; and he has the proper "need to know" in reference to the object.

**ADP (Automatic Data Processing)**

An assembly of computer equipment, facilities, personnel, software and procedures configured for the purpose of classifying, sorting, calculating, computing, summarizing, storing, and retrieving data and information with a minimum of human intervention.

**anonymous user**

An anonymous user is an unregistered user of the Multics system whose personid (see below) is "anonymous"; in other words, his personid is unknown to the system. An anonymous user may or may not be required to furnish a password in order to gain access to the system.

**branch**

A branch is a component of a directory which describes an immediately inferior segment or directory.

**breach**

The successful and repeatable defeat of security controls with or without an arrest which, if carried to consummation, could result in a penetration of the system. Examples of breaches are:

Operation of user code in master mode;

Unauthorized acquisition of userid and password; and

Accessing a file without using prescribed operating system mechanisms.

**category set (also see access)**

In reference to a person, a category set refers to the set of compartments a person is eligible to access. (The maximum number of compartments within a single system is limited to sixteen.) A compartment in this context is an orthogonal subdivision of the classification levels. A compartment is like a formal need to know authorization to information of a certain topic without consideration of classification level.

In reference to documents, files or other objects, a category set refers to the possible information sources used

to create the object. Thus a category set with several categories, or compartments, would indicate that the object should be handled with the extra caution accorded to objects which would intersect the sensitive areas of each of the categories in the set.

#### classification (also see access and security level)

One of an ordered set of levels which describes the sensitivity of the information to which it refers. (The maximum number of levels within a single system is limited to seven.) Only information, documents, data, equipment or other objects have classification levels. Persons need the correct clearance to access information at any level higher than Unclassified.

When compartmented security is used, a classification includes both the level and the category set associated with an object.

#### clearance (also see access and security level)

The eligibility of a person (or process) to access information of a certain classification level (or lower). For example, a person with a Secret clearance is eligible to access information with classification levels Unclassified to Secret, but may not have access to Top Secret information.

When compartmented security is used, a clearance also includes the categories a person is eligible to access.

In addition to the eligibility afforded a person by his clearance, he must also have the need to know the classified information before he is given access.

#### daemon (SysDaemon)

Certain Multics processes are dedicated to performing supervisory functions, such as handling I/O requests and backing up the storage system. These processes are called daemon processes and run with the SysDaemon projectid.

#### directory

A directory is a segment in the Multics storage system hierarchy maintained by the supervisor which contains information about immediately inferior segments. A

directory contains a list of branches analogous to a table of contents.

#### fixed level property

In order that no breach of security occur within the computer system environment, it is sufficient that no process be permitted to read information classified above its clearance, nor be permitted to write information classified below its clearance. This principle is known as the fixed level property. (It has also been called the "\*-property" in some of the referenced literature)

#### initializer process (system control process, answering service)

The initializer process is a special process which performs certain system-controlling functions. In particular, it initializes the Multics environment, monitors and allocates terminals, creates all other processes, and performs accounting functions.

#### interprocess communication (ipc)

Interprocess communication is a facility which allows one process to communicate with another in a controlled manner. Both the sending and receiving processes must adhere to a specified protocol.

#### Multi-Level Security Mode

A mode of operating under an operating system (supervisor or executive program) which provides a capability permitting various levels and categories or compartments of material to be concurrently stored and processed in an ADP System. In a remotely accessed resource-sharing system, the material can be selectively accessed and manipulated from terminals by personnel having different security clearances and access approvals. This mode of operation can accommodate the concurrent processing and storage of (a) two or more levels of classified data, or (b) one or more levels of classified data with unclassified data depending upon the constraints placed on the systems by the Designated Approving Authority.

**Operating System (O/S)**

An integrated collection of service routines for supervising the sequencing and processing of programs by a computer. Operating systems control the allocation of resources to users and their programs and play a central role in assuring the secure operation of a computer system. Operating systems may perform debugging, input-output, accounting, resource allocation, compilation, storage assignment tasks, and other system related functions (Synonymous with Monitor, Executive, Control Program, and Supervisor).

**personId**

The registered name of someone who is authorized to use the system. It is usually constructed from the last name (surname) of the person.

**process**

A process is the active agent of the user on Multics and (in the security system) has a clearance which may not exceed the user's clearance. The lifetime of a process normally corresponds to a user's terminal session and is described internally by an address space and a point of execution. Both the address space and the execution point are dynamic over the life of the process.

**projectId**

The registered name of a project which has an account on the system.

**Remotely Accessed Resource-Sharing Computer System**

A computer system which includes one or more central processing units, peripheral devices, remote terminals, and communications equipment or interconnection links, which allocates its resources to one or more users, and which can be entered from terminals located outside the central computer facility.

**security level (see also Clearance and Classification)**

This term is used frequently as an abbreviation for the level/category combination which describes a clearance or a classification. Thus the "level" of a process is the clearance of the process and the "level" of a segment is the classification of the segment.

**segment**

A segment is a logical unit of storage on Multics. It roughly corresponds to a file stored on a disk pack and accessible to a user. The segment is the smallest element of supervisor access control in the Multics storage system.

**Trojan Horse**

A Trojan Horse is a procedure which provides a potentially useful function to attract use by a person having access privileges not possessed by the author of the procedure. The Trojan Horse program detects such use and performs unauthorized or unwanted functions which would allow the author of the procedure to obtain information to which he did not otherwise have access or to perform acts of sabotage which would not otherwise be possible.

**user**

An instance of a person logged into the system on a project. A user is identified by a userid.

**userid**

A table entry which would describe a user (e.g. an access control list entry). A userid consists of "personid.projectid.tag," where tag is normally "a" for an interactive user, "m" for an absentee user, and "z" for certain system daemons. The userid is also called the "principal identifier" or "group\_id" of the user.

**3.1.4 Security Access Rules - The Model**

Access control is generally described as a subject attempting to access an object through an intervening reference monitor. The reference monitor checks, each and every time a subject attempts to access an object, to see if the subject has the proper authorization to perform the

desired operation (e.g. read, write, execute, append, modify, delete). In Multics, a process is the only subject which can make a reference to any object. The set of objects are segments, directories, branches, I/O devices, message segments, message segment messages and interprocess communication messages. Each object has a classification level and category set associated with it (i.e., a security level.)

In Multics, the reference monitor which validates each reference to an object is the "ring 0" supervisor in conjunction with processor hardware protection mechanisms. Within the protection ring scheme supported by the Honeywell 68/80 processor, ring 0 is the most privileged and most protected ring of operation. All access control decisions are made within ring 0. Each time a process attempts to gain access to an object, the clearance of the process is compared with the classification of the object and access is either granted or denied in accordance with rules designed to emulate the military security system. In addition to classification, certain objects such as segments and directories have an associated access control list which specifies persons having need to know authorization as in the military security system.

When the security level of two objects is compared, four relationships are possible:

less than

equal

greater than

isolated

The security level of object 1 is considered "less than" the security level of object 2 if:

1. The level of object 1 is numerically less than or equal to the level of object 2; and
2. The category set of object 1 is a subset of the category set of object 2; and
3. The security level of object 1 is not equal to the security level of object 2.

The security levels of two objects are considered "equal" if:

1. The levels are numerically equal; and
2. The category sets are identical.

The security level of object 1 is considered "greater than" the security level of object 2 if:

1. The level of object 1 is numerically greater than or equal to the level of object 2; and
2. The category set of object 2 is a subset of the category set of object 1; and
3. The security level of object 1 is not equal to the security level of object 2.

The security levels of two objects are considered "isolated" if the category sets are isolated.

The "minimum" of several security levels is defined as:

1. The numerical minimum of the levels; and
2. The intersection of the category sets.

In order for a person to access information, the military security system requires that the clearance of the person be greater than or equal to the classification of the information. A sufficient condition for satisfying this requirement within the computer system environment is the enforcement of the following two rules:

1. A process having clearance  $\Delta$  may not "read up," i.e. read an object having a classification greater than  $\Delta$ .
2. A process having clearance  $\Delta$  may not "write down," i.e. write an object having a classification less than  $\Delta$ .

With these two rules enforced, it is impossible for any process to extract information from an object of higher classification or to transfer information from an object of higher classification to an object of lower classification. Hence, no compromise of classified information can occur. This principle is known as the "fixed level property."

It is important to recognize that the rules described above represent a sufficient, but not a necessary condition for achieving security. Although the fixed level property restrictions will be strictly enforced for all user processes, they will, in certain circumstances, be applied interpretively for trusted system processes. In no circumstances, however, will security be violated, because

trusted system processes ~~must~~ operate correctly.

## 3.2 Overview of System Architecture

### 3.2.1 Hardware Configuration

Information transmitted between hardware modules must be carefully controlled by the system and no user should be able to directly affect the action of an active module (except for the CPU).

### 3.2.2 Internal/External Environment

The system can be logically divided into two environments: internal and external. The internal environment is totally controlled by the system. This includes: processors, memory, disk drives, I/O multiplexers, bulk store, communication processors, and tape drives used for system functions.

The external environment can be directly influenced by the actions of a process. This environment includes: terminals, line printers, card readers, card punches, non-system tape drives, and other devices in the I/O class not used for system functions.

### 3.2.3 The Security Kernel

(To be determined at a later date.)

### 3.2.4 The Secure Front End Processor

(To be determined at a later date.)

### 3.2.5 The System Security Perimeter

Honeywell

Draft 1-31-76

(To be determined at a later date.)

### 3.3 Process Creation

The Multics access control mechanism depends on several important factors. First and foremost is the notion of an unforgeable "user name" and clearance which identifies the access rights of a Multics process; the entity which performs all tasks on behalf of the human user. A Multics "user name" consists of three components: Person, Project, and Tag. The Person component uniquely identifies a registered user of Multics. The Project component identifies a registered project, and Tag is presently derived from the type of process (i.e. interactive, absentee, or consoleless daemon).

#### 3.3.1 Logging Into the System

All interactive users of Multics must login from a terminal. Terminal identification plays an important role in determining the classification of data which can be accessed by a process. Clearly we cannot allow a user to extract sensitive data while using a dial up terminal connection from a phone booth in a public area, even though he is properly cleared. Terminals are associated with the security level of the controlled area in which they are located. Each terminal (or multiplex group as in a network) has a security level which describes the highest classification of data which can be processed from the terminal. Physical control of access to the terminal area ensures that all persons who could see the terminal output are cleared to at least the security level of the terminal. This will be discussed further in Section 3.3.3.

There are two classes of interactive users on Multics: registered users and anonymous users.

A registered user is known to the system by his name (personid) and is associated with one or more projects (projectid) for accounting and resource control purposes. Each personid has a password to authenticate his identity.

To log into Multics, a registered user must give his personid and projectid to the system control process. Control arguments to the login command allow the user to

specify a desired security level for his process, as well as other attributes. The password will be requested immediately following the login command.

Anonymous users should not normally be permitted on the system since password authentication is not always required for them. Where passwords are required for anonymous users, these passwords are controlled by project administrators rather than the system administrator (SA) or the system security administrator (SSA). If, at any time, anonymous users are permitted on the system, they are always be assigned system\_low processes.

### 3.3.2 User Authentication

In order for Multics to successfully enforce access controls, it must be possible to uniquely and positively identify each user at login. This is presently accomplished by assigning each registered person his own password, and at each login, requesting his password for verification purposes. If the password stored by Multics matches the password given by the user, Multics assumes the user is valid, and creates a process with the "user name" (userid) of the user. If, after giving the user several chances (to allow for typing mistakes), a correct password has not been received, Multics refuses the login.

Clearly, the password is a vital part of the access control mechanism, and as such, must be carefully protected by both the user and the system. If a person could guess (by whatever means) another user's password, that person would himself be able to log in as the other user. A person who learns another person's password will not be able to log in with the same clearance as the owner of the password unless he, himself, has an equal or higher clearance which affords him access to a terminal of equal or higher classification. Therefore, password compromise can, at worst, result in sabotage or need to know violations.

An attempted login may be rejected for the following reasons:

1. illegal login word
2. incorrect personid or projectid
3. incorrect password

4. incorrect security level
5. unrecognized login option

These rejected login attempts are recorded for audit purposes. In addition, if a user attempts to use a terminal with a maximum clearance greater than the personid clearance, a message is sent to the operator, since this will indicate a breach of physical security.

### 3.3.3 Process Clearance Assignment

When a process is created for a user, a clearance is established for the process. This clearance is not changeable for the life of the process. It is the process clearance which is used to determine a user's authorization to access classified information in the system.

The data associated with a personid (the system unique identification for the person) contains the clearance of the person. Similar clearance data is associated with each projectid. In addition, the data which describes the limitations of a person on a given project contains clearance data.

The clearance to be assigned to a process is determined as follows:

1. No process will be created for a given userid. i.e. a given person on a given project, with a higher clearance than the minimum of the person's clearance, the project's clearance, and the person's clearance within the project.
2. No user should be able to create a process with a higher clearance than the maximum clearance of his terminal.
3. A user may request a process with a lower clearance than the minimum of his userid and terminal clearances.
4. A user is able to specify a default login clearance (no higher than his personid clearance).

Only the SSA is able to assign clearances for a personid or a projectid. If the SSA lowers the clearance of a personid, the user's process is forceably terminated if the user has an active process with a clearance greater than the downgraded clearance of the personid.

The Channel Definition Table (CDT) is used by the system control process to check the maximum clearance of the terminal being used by a person attempting to log in. Normally terminals will be "hard wired" to the system, thereby allowing each terminal to be uniquely identified by an associated channel number. In the general case, there may be crypto-dial-up terminals. However, in that case, the crypto units will provide the unique terminal identification. As an extra check, the answerback code received from a terminal is compared against its "registered" answerback code. This answerback test is useful in detecting mistakes, as well as malicious tampering, involving communications lines and terminals.

When a process is created, there are two security levels assigned to it: the current authorization and the max\_authorization. The current authorization is the security level of the process that is used to control access to stored information. The max\_authorization is the highest authorization that can be assigned to the process userid.

The max\_authorization is calculated as the minimum of the clearances for the personid, the projectid, and the person-project combination.

The current authorization is calculated as the minimum of the max\_authorization, the security level of the terminal and the requested security level (from the login option or default security level.)

Each user is told his process clearance at the beginning of the process. In this way, the user is made explicitly aware of his level of operation. Hence, mistakes such as placing Top Secret information in a Secret file are unlikely to occur.

Absentee processes are created at the level of the requesting process. A user is not able to create an absentee process with a security level which is lower than that of his current process, since the passing of arguments to the absentee process would constitute a write-down operation. The ability for a user to enter an absentee request of a higher security level than his process clearance is one way for a Trojan Horse to gain control of a user's access permissions. If this happens, a need to know violation or sabotage can occur. Therefore, absentee processes are restricted to the security level of the requesting process.

A new\_proc option allows a user to upgrade/downgrade his security level. When no option is specified, the default security level for the new process will be that of the

current process. (The same will be true for abnormal termination of a process).

#### 3.3.4 Potential Security Problems

By providing a means for a user to change his security level through program control (new\_proc -with level option), a Trojan Horse could set itself up as the program to be called when a user attempts to change to a new security level. An elaborate Trojan Horse could totally simulate system action for new\_proc to fool the user into thinking he is operating at a higher level. Now if the user attempts to input classified data, the Trojan Horse could, by simulating the entire user interface, cause the user to put the classified data into a segment with a lower classification. This problem can be solved by only allowing a user to "new\_proc" to the same or lower security level.

In a similar manner, a user may write his own "logout -hold" command to fool the next user of the terminal into thinking he is talking to the system instead of the previous user's process. This could allow a malicious user to capture the password of another user, thus permitting sabotage and need to know violations. Also, the user environment simulation described above could be used here. The solution to this problem is to require the terminal to be powered off by each user before attempting to login. (This can be handled several ways. The choice is up to the site manager.)

Solutions exist to all of the above potential problems. However, given the low expectation of occurrence of these problems, the required sacrifices in user convenience were felt to be unwarranted.

### 3.4 The Multics Storage System

The individual user is able to specify which users should have "need to know" for a given segment or directory by use of the Access Control List. The mode of access (e.g. read, write) allowed to a process by the current Multics Access Control List is further restricted to ensure compliance with the fixed level property rules. In other words, the fixed level property rule takes precedence over the Access Control List.

#### 3.4.1 Access to Segments

(To be determined at a later date.)

#### 3.4.2 Access to Directories

(To be determined at a later date.)

3.5 Use of I/O Devices

(To be determined at a later date.)

3.5.1 Internal I/O

(To be determined at a later date.)

3.5.2 External I/O

(To be determined at a later date.)

3.5.3 Bulk I/O Services

(To be determined at a later date.)

**3.6 Interprocess Communications**

(To be determined at a later date.)

**3.6.1 Block/Wakeup Mechanism**

(To be determined at a later date.)

**3.6.2 Message Segments**

(To be determined at a later date.)

**3.7 Trusted Processes and System Functions**

(To be determined at a later date.)

**3.7.1 System Control Process**

(To be determined at a later date.)

**3.7.2 Backup**

(To be determined at a later date.)

**3.7.3 Retrieval**

(To be determined at a later date.)

**3.7.4 The System Security Administrator**

(To be determined at a later date.)

**3.7.5 The System Administrator**

(To be determined at a later date.)

**3.7.6 Maintenance and Repair Processes**

(To be determined at a later date.)

**3.7.7 The I/O Coordinator**

(To be determined at a later date.)

3.8 System Audit

(To be determined at a later date.)

4.0 Quality Assurance

This section does not apply to this report.

## 5.0 Preparation for Delivery

This contract only calls for the preparation of reports describing the development of a prototype secure Multics and the demonstration of its characteristics.

None of the hardware modifications described in this specification or the software modifications to be implemented during the development of a prototype secure Multics are to be formally delivered or supported by Honeywell.

## 6.0 Notes

This section contains information which is not contractually binding. It will be filled in as various side issues of the project are identified.